IN THE CLAIMS:

1. (Currently Amended) A method for operating a data storage system, comprising:

creating a writable virtual disk (vdisk) at a selected time, the writable vdisk referencing changes in data stored in the data storage system after the writable vdisk was created;

maintaining a backing store, the backing store referencing data stored in the data storage system which has not been changed since the writable vdisk was created;

loading blocks of the writable vdisk into a memory, the loaded writable vdisk blocks including a writable vdisk indirect block having a plurality of fields, each writable vdisk indirect block field storing a valid pointer to a writable vdisk data block or an invalid pointer representing one of a plurality of holes, where each hole instructs the data storage system to examine a corresponding virtual block number pointer in the backing store;

loading blocks of the backing store into the memory, the loaded backing store blocks including a backing store indirect block having a plurality of fields, each backing store indirect block field corresponding to a ~~field of the~~ writable vdisk indirect block field, one or more backing store indirect block fields having a pointer to a backing store data block;

searching each field of the writable vdisk indirect block for a hole; and

filling each hole in the ~~writeable~~ writable vdisk by replacing each invalid pointer with a pointer to the backing store data block referenced by the corresponding backing store indirect block field to update the writable vdisk to reference both the data which is unchanged since the writable vdisk was created and the data which has been changed since the writable vdisk was created.

2. (Currently Amended) The method of claim 1, further comprising:

dirtying the backing store data block pointed to by the backing store indirect block to enable write allocation of the dirty backing store data block without altering a data content of the backing store data block.

3. (Currently Amended) The method of claim ~~1~~2, further comprising:
choosing a new pointer for a newly allocated data block containing an unaltered data content;
setting bits in block allocation structures for the newly allocated data block; and
placing the new pointer to the newly allocated data block into the field of the writable vdisk indirect block to replace the hole.

4. (Currently Amended) The method of claim 3 further comprising:
freeing the dirty backing store data block; and
writing the newly allocated data block to disk.

5. (Currently Amended) The method of claim 4 further comprising:
releasing an association of the writable vdisk to the backing store to thereby separate the writable vdisk ~~data~~ block~~s~~ from the backing store ~~data~~ blocks.

6. (Original) The method of claim 1 wherein the pointers contained in the writable vdisk indirect block fields and the backing store indirect block fields comprise logical volume block numbers (VBNs).

7. (Currently Amended) The method of claim 1 wherein the invalid pointer~~s~~ contained in the writable vdisk indirect block field~~s~~ comprises a zero logical volume block number (VBN).

8. (Original) The method of claim 1 wherein the plurality of fields in the writable vdisk indirect block are a writable vdisk level 1 buffer and the plurality of fields in the backing store indirect block are a backing store level 1 buffer.

9. (Currently Amended) An apparatus for operating a computer database, comprising:

a writable virtual disk (vdisk) created at a selected time, the writable vdisk referencing changes in data stored in a data storage system after the writable vdisk was created;

a backing store, the backing store referencing data stored in the data storage system which has not been changed since the writable vdisk was created;

a backdoor message handler that loads blocks of the writable vdisk and backing store from disk into a memory of the storage system;

a writable vdisk indirect block in the memory having a plurality of fields, each writable vdisk field storing a valid pointer to a writable vdisk data block or an invalid pointer representing one of a plurality of holes, where each hole instructs the data storage system to examine a corresponding virtual block number pointer in the backing store;

a backing store indirect block in the memory having a plurality of fields, each backing store indirect block field corresponding to a ~~field of the~~ writable vdisk indirect block field, each backing store indirect block field having a pointer to a backing store data block;

a special loading function that searches each field of the writable vdisk indirect block for one or more fields representing a hole; and

a write allocator that fills each hole in the ~~writeable~~ writable vdisk by replacing each invalid pointer with a pointer to the backing store data block referenced by the corresponding backing store indirect block field to update the writable vdisk to reference both the data which is unchanged since the writable vdisk was created and the data which has been changed since the writable vdisk was created.

10. (Currently Amended) The apparatus of claim 9 wherein the write allocator further ~~comprises:~~ chooses a new pointer for a newly allocated data block containing unaltered data content, set bits in block allocation structures for the newly allocated data block, and place the new pointer to the newly allocated data block into the field of the writable vdisk indirect block to replace the hole.

11. (Currently Amended) The apparatus of claim 10 wherein the write allocator further frees the ~~dirty~~ backing store data block and writes the newly allocated data block to disk.

12. (Currently Amended) The apparatus of claim 9 wherein the backdoor message handler further loads the blocks of the writable vdisk and the blocks of the backing store during periods of reduced processing activity.

13. (Currently Amended) The apparatus of claim 9 wherein the pointers contained in the writable vdisk indirect block fields and ~~the~~ pointers in backing store indirect block fields comprise logical volume block numbers (VBNs).

14. (Currently Amended) The apparatus of claim 9 wherein the invalid pointer~~s~~ contained in the writable vdisk indirect block field~~s~~ comprise a zero logical volume block number (VBN).

15. (Currently Amended) The apparatus of claim 9 wherein the plurality of fields in the writable vdisk indirect block comprise~~s~~ a writable vdisk level 1 buffer and the plurality of fields in the backing store indirect block comprise~~s~~ a backing store level 1 buffer.

16.-18. (Cancelled).

19. (Currently Amended) A data storage system apparatus, comprising:

means for creating a writable virtual disk (vdisk) at a selected time, the writable vdisk referencing changes in data stored in the data storage system after the writable vdisk was created;

means for maintaining a backing store, the backing store referencing data stored in the data storage system which has not been changed since the writable vdisk was created;

means for loading blocks of the writable vdisk from a disk into a memory, the loaded writable vdisk blocks including a writable vdisk indirect block having a plurality of fields, each writable vdisk field storing a valid pointer to a writable vdisk data block or an invalid pointer representing one of a plurality of holes, where each hole instructs the data storage system to examine a corresponding virtual block number pointer in the backing store;

means for loading blocks of the backing store from a disk into the memory, the loaded backing store blocks including a backing store indirect block having a plurality of fields, each backing store indirect block field corresponding to a ~~field of the~~ writable vdisk indirect block field, one or more backing store indirect block fields having a pointer to a backing store data block;

means for searching each field of the writable vdisk indirect block for a hole; and

means for filling each hole in the writable vdisk by replacing each invalid pointer with a pointer to the backing store data block referenced by the corresponding backing store indirect block field to update the writable vdisk to reference both the data which is unchanged since the writable vdisk was created and the data which has been changed since the writable vdisk was created .

20. (Currently Amended) A non-transitory computer readable medium executable program instructions executed by a processor, comprising:

program instructions that create a writable virtual disk (vdisk) at a selected time, the writable vdisk referencing changes in data stored in a data storage system after the writable vdisk was created;

program instructions that maintain a backing store, the backing store referencing data stored in the data storage system which has not been changed since the writable vdisk was created;

program instructions that load blocks of the writable vdisk from a disk into a memory, the loaded writable vdisk blocks including a writable vdisk indirect block having a plurality of fields, each writable vdisk field storing a valid pointer to a writable vdisk data block or an invalid pointer representing one of a plurality of holes, where each hole instructs the data storage system to examine a corresponding virtual block number pointer in the backing store;

program instructions that load blocks of the backing store from a disk into the memory, the loaded backing store blocks including a backing store indirect block having a plurality of fields, each backing store indirect block field corresponding to a ~~field of the~~ writable vdisk indirect block field, one or more backing store indirect block fields having a pointer to a backing store data block;

program instructions that search each field of the writable vdisk indirect block for a hole; and

program instructions that fill each hole in the ~~writeable~~ writable vdisk by replacing each invalid pointer with a pointer to the backing store data block referenced by the corresponding backing store indirect block field to update the writable vdisk to reference both the data which is unchanged since the writable vdisk was created and the data which has been changed since the writable vdisk was created.

21.-22. (Cancelled).

23. (Currently Amended) A method for operating a data storage system, comprising:

creating a writable virtual disk (vdisk) at a selected time, the writable vdisk referencing changes in data stored in the data storage system after the writable vdisk was created, the writable vdisk having a plurality of holes where each hole instructs the storage system to examine a corresponding virtual block number pointer in a backing store;

maintaining the backing store, the backing store referencing the data stored in the data storage system which has not been changed since the writable vdisk was created;

searching each field of the writable vdisk for a hole; and

filling each hole in the writable vdisk by replacing each hole in the writable vdisk to point to ~~the~~ a data block referenced by ~~the~~ a corresponding backing store indirect block of the backing store to fill each hole of the ~~writeable~~ writable vdisk with the data block referenced by the corresponding backing store indirect block and thus update the writable vdisk to reference both the data which is unchanged since the writable vdisk was created and the data which has been changed since the writable vdisk was created.

24. (Previously Presented) The method of claim 23, further comprising:
dirtying the data block pointed to by the backing store indirect block to enable write allocation of the dirty data block without altering a data content of the data block.

25. (Currently Amended) The method of claim ~~23~~ 24 further comprising:
choosing a new pointer for a newly allocated data block containing an unaltered data content;
setting bits in block allocation structures for the newly allocated data block; and
placing the new pointer to the newly allocated data block into ~~the~~ a field of ~~the~~ a writable vdisk indirect block to replace the hole.

26. (Previously Presented) The method of claim 25, further comprising:
freeing the dirty data block; and
writing the newly allocated data block to disk.

27. (Currently Amended) The method of claim 26 further comprising:
releasing an association of the writable vdisk to the backing store to thereby separate ~~the~~ writable vdisk ~~data~~ blocks from ~~the~~ backing store ~~data~~ blocks.

28. (Currently Amended) The method of claim 23, further comprising:
including logical volume block numbers (VBNs) in ~~the~~ pointers contained in ~~the~~ a writable vdisk indirect block field~~s~~ of the writable vdisk and ~~the~~ a backing store indirect block field~~s~~ of the backing store.

29. (Currently Amended) The method of claim ~~23~~28, further comprising:
using a zero logical volume block number (VBN) as ~~the~~ an invalid pointer~~s~~ contained in the hole of the writable vdisk indirect block field~~s~~.

30. (Currently Amended) The method of claim 23, further comprising:

using a writable vdisk level 1 buffer for ~~the~~ a plurality of fields in ~~the~~ a writable vdisk indirect block of the writable vdisk and using a backing store level 1 buffer for ~~the~~ a plurality of fields in the backing store indirect block.

31. (Currently Amended) A data storage system, comprising:

a writable virtual disk (vdisk) created at a selected time, the writable vdisk referencing changes in data stored in the data storage system after the writable vdisk was created, the writable vdisk having a plurality of holes , each hole instructing the storage system to examine a corresponding virtual block number pointer in a backing store;

the backing store referencing data stored in the data storage system which has not been changed since the writable vdisk was created;

a processor that searches each field of the writable vdisk for a hole; and

the processor that fills each hole in the writable vdisk so that each hole in the writable vdisk points to ~~the~~ a data block referenced by ~~the~~ a corresponding backing store indirect block to fill each hole of the ~~writeable~~ writable vdisk with the data block referenced by the corresponding backing store indirect block and thus update the writable vdisk to reference both the data which is unchanged since the writable vdisk was created and the data which has been changed since the writable vdisk was created.

32. (Currently Amended) The system of claim 31, ~~further comprising:~~wherein the processor further dirties the data block pointed to by the backing store indirect block ~~are dirtied~~ to enable write allocation of the dirty data block without altering a data content of the data block.

33. (Currently Amended) The system of claim ~~31~~32 ~~further comprising:~~wherein the processor further:

chooses a new pointer ~~chosen~~ for a newly allocated data block containing an unaltered data content;

sets bits ~~are set~~ in ~~a~~ block allocation structures for the newly allocated data block; and places a new pointer to the newly allocated data block ~~placed~~ into a field of the writable vdisk indirect block to replace the hole.

34. (Currently Amended) The system of claim 33, ~~further comprising~~wherein the processor further:

frees the dirty data block ~~is freed~~; and

writes the newly allocated data block ~~is written~~ to disk.

35. (Currently Amended) The system of claim 34 ~~further comprising:~~wherein the processor further releases an association of the writable vdisk to the backing store ~~is released~~ to thereby separate ~~the~~ writable vdisk ~~data~~ blocks from ~~the~~ backing store ~~data~~ blocks.

36. (Currently Amended) The system of claim 31, ~~further comprising:~~wherein the processor further has logical volume block numbers (VBNs) included in ~~the~~ pointers contained in ~~the~~ a writable vdisk indirect block field~~s~~ of the writable vdisk and ~~the~~ a backing store indirect block field~~s~~ of the backing store.

37. (Currently Amended) The system of claim ~~31~~36, ~~further comprising:~~ wherein the processor further uses a zero logical volume block number (VBN) ~~used~~ as ~~the~~ an invalid pointer~~s~~ in the hole contained in the writable vdisk indirect block field~~s~~.

38. (Currently Amended) The system of claim 31, ~~further comprising:~~wherein the processor further uses a writable vdisk level 1 buffer ~~used~~ for ~~the~~ a plurality of fields in ~~the~~ a writable vdisk indirect block of the writable vdisk and use a backing store level 1 buffer ~~used~~ for ~~the~~ a plurality of fields in the backing store indirect block.

39. (Currently Amended) A non-transitory computer readable medium containing executable program instructions executed by a processor comprising:

program instructions that create a writable virtual disk (vdisk) at a selected time, the writable vdisk referencing changes in data stored in the data storage system after the writable vdisk was created, the writable vdisk having a plurality of holes where each hole instructs the storage system to examine a corresponding virtual block number pointer in a backing store;

program instructions that maintain the backing store, the backing store referencing data stored in the data storage system which has not been changed since the writable vdisk was created;

program instructions that search each field of the writable vdisk for a hole; and

program instructions that fill each hole in the writable vdisk to point to ~~the~~ a data block referenced by ~~the~~ a corresponding backing store indirect block to fill each hole of the ~~writeable~~ writable vdisk with the data block referenced by the corresponding backing store indirect block and thus update the writable vdisk to reference both the data which is unchanged since the writable vdisk was created and the data which has been changed since the writable vdisk was created.

40. (Currently Amended) A method for operating a data storage system, comprising:

creating a writable virtual disk (vdisk) at a selected time, the writable vdisk referencing changes in data stored in the data storage system after the writable vdisk was created, the writable vdisk having a plurality of holes where each hole instructs the data storage system to examine a corresponding virtual block number pointer in a backing store;

maintaining the backing store, the backing store referencing the data stored in the data storage system which has not been changed since the writable vdisk was created;

searching, by a background task process, each field of the writable vdisk for a hole;

for each hole in the ~~writeable~~ writable vdisk, marking as dirty ~~the~~ a corresponding data block pointed to by ~~the~~ a backing store indirect block without modifying the corresponding data block; and performing a write allocation to replace each hole in the writable vdisk to point to the data block marked as dirty and referenced by the corresponding backing store indirect block to update the writable vdisk to reference both the data which is unchanged since the writable vdisk was created and the data which has been changed since the writable vdisk was created.

41. (Currently Amended) A data storage system, comprising:

a writable virtual disk (vdisk) created at a selected time, the writable vdisk referencing changes in data stored in the data storage system after the writable vdisk was created, the writable vdisk having a plurality of holes where each hole instructs the data storage system to examine a corresponding virtual block number pointer in the backing store;

the backing store referencing the data stored in the data storage system which has not been changed since the writable vdisk was created;

a background task processor that searches each field of the writable vdisk for a hole; and

the background task processor that marks as dirty, for each hole in the ~~writeable~~ writable vdisk, ~~the~~ a corresponding data block pointed to by ~~the~~ a backing store indirect block without modifying the corresponding data block, and performs a write allocation to replace each hole in the writable vdisk to point to the data block marked as dirty and referenced by the corresponding backing store indirect block to update the writable vdisk to reference both the data which is unchanged since the writable vdisk was created and the data which has been changed since the writable vdisk was created.

42. (Currently Amended) A non-transitory computer readable medium containing executable program instructions executed by a processor, comprising:

program instructions that create a writable virtual disk (vdisk) at a selected time, the writable vdisk referencing changes in data stored in the data storage system after the writable vdisk was created, the writable vdisk having a plurality of holes where each hole instructs the data storage system to examine a corresponding virtual block number pointer in a backing store;

program instructions that maintain the backing store, the backing store referencing the data stored in the data storage system which has not been changed since the writable vdisk was created;

program instructions that search, by a background task process, each field of the writable vdisk for a hole;

program instructions that mark as dirty, for each hole in the ~~writeable~~ writable vdisk, ~~the~~ a corresponding data block pointed to by ~~the~~ a backing store indirect block without modifying the corresponding data block; and

program instructions that perform a write allocation to replace each hole in the writable vdisk to point to the data block marked as dirty and referenced by the corresponding backing store indirect block to update the writable vdisk to reference both the data which is unchanged since the writable vdisk was created and the data which has been changed since the writable vdisk was created.